ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

REPORT ON FINAL PROJECT

CONTENT-BASED CHARACTER RETRIEVAL FROM MOVIES

Subject: Advanced Computer Vision

Class: CS331.O11.KHCL

Instructor: PhD. Mai Tien Dung

Members: Do Minh Khoi - 21521007

Nguyen Nguyen Khoi - 21521009

HO CHI MINH CITY, 2024

Contents

I. INTRODUCTION

1.1. Content-based video retrieval

- Information retrieval is the process of obtaining information system resources that are relevant to an information need from a collection of those resources. Resources can be of text (books, journals, etc.), audio (music, speeches, etc.), video (movie, recordings, etc.)
- Huge amount of data is generated on a daily basis. But this data is actually useless, if there is no way to obtain and query the data, the information we collect is useless. Information retrieval system is critical for making sense of data. Without Google or any other search engines, it would have been very difficult to retrieve any information off the Internet.
- Content-based video retrieval (CBVR) is a fascinating technology that allows you to search for videos in a database based on their actual content, rather than just relying on text tags or metadata. CBVR systems first analyze the videos in the database, extracting various features from each one. These features can be visual, audio, or textual features. They are then used to create an index for each video. When users submit a query, the system analyzes the features of your query (which could be another video, image or text) and calculates its similarity to the indexed videos. Finally, the system ranks the videos in the database based on their similarity to the query. The most similar videos are presented to the users as the search results.
- Our project is **"Content-based character retrieval from movies"**. We construct indexes for each movie, then we build a system to get input as a movie and query images of a character in that movie and provide output as movie shots related to the query images. To search for the character in the query, we analyze different pretrained models to extract features from query and from frames in the movie.

1.2. Some concepts regarding video processing

1.2.1. Scene

- The largest unit in a video, representing a continuous sequence of shots filmed at a single location or under the same setting. Think of it as a complete event within the video, like a conversation in a park or a car chase through the city.

1.2.2. Shot

- A continuous video clip captured by a single camera without interruption. Imagine each shot as a distinct camera angle or perspective within a scene.

1.2.3. Frame

- The smallest unit in a video, representing a single captured image at a specific point in time. A video is essentially a rapid sequence of frames played back to create the illusion of motion. Think of each frame as a still photograph frozen in time, capturing a moment within a shot.
- Frame rate is typically the frequency (rate) at which consecutive images (frames) are captured, displayed or cut in a portion of time. The most common unit for measuring frame rate is frames per second (FPS). Cinematic movies are usually displayed at 24 FPS, since this frame rate is similar to how we see the world and creates a very real look. However, when cutting these movies into frames in this report, we stick to 5 FPS, which is the most common frame rate used in video processing.

1.3. Content-based character retrieval from movies

1.3.1. Input

- A database (index) of a movie
- A set of query images of a character in that movie

1.3.2. Output

- All shots in which the character appear (considered within the movie in the input)
- A set of query images of a character in that movie

1.3.3. Constraint

- Not a cartoon movie

1.3.4. Requirements

- The result must cover all true shots, including ones in which the person is not clear (faces are unable to detect, too much illumination, etc.)
- As many relevant shots retrieved as possible

1.3.5. Applications

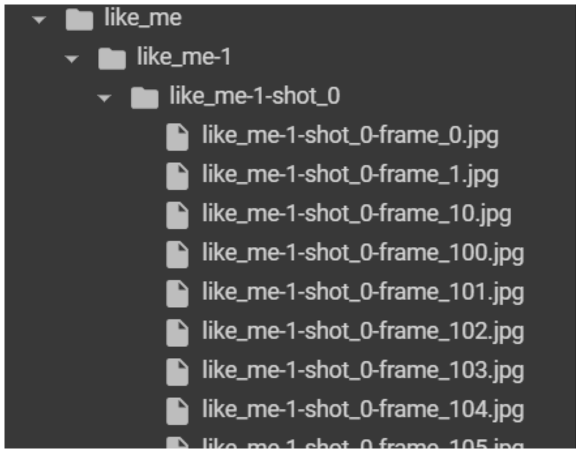
- Quickly rate a character's performance without watching the entire movie
- Plays a crucial role in key-event summarization task: retrieving important shots from a movie

II. METHODS

2.1. Pipeline of the retrieval system

2.1.1. Offline stage

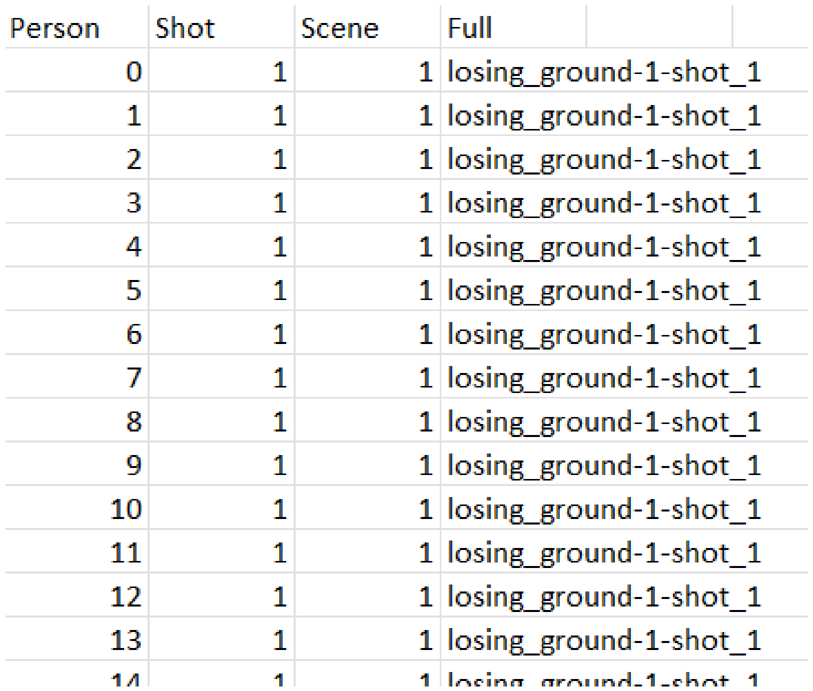
- Extract all scenes, shots and frames of a movie; in each scene, for each shot, save corresponding frames in a folder
- An example of the organization of scenes, shots, and frames in Like Me film is shown below. The movie contains many folders, each folder is a scene named {movie_name}-{scene no.}. The folder again contains many other folders, each folder is a shot named {movie_name}-{scene no.}-shot_{shot no.}. Each shot's folder contains many frames in form of .jpg images. Each image is named {movie_name}-{scene no.}-shot_{shot no.}-frame_{frame no.}.jpg



```
like_me
    like_me-1
        like_me-1-shot_0
            like_me-1-shot_0-frame_0.jpg
            like_me-1-shot_0-frame_1.jpg
            like_me-1-shot_0-frame_10.jpg
            like_me-1-shot_0-frame_100.jpg
            like_me-1-shot_0-frame_101.jpg
            like_me-1-shot_0-frame_102.jpg
            like_me-1-shot_0-frame_103.jpg
            like_me-1-shot_0-frame_104.jpg
            like_me-1-shot_0-frame_105.jpg
```

Example of the organization of a movie

- Detect persons from frames
- Extract and normalize feature vectors of these persons
- Construct a database (more precisely an indexing) to store these feature vectors
- Map the order of each vector (person) in the database to its shot in a .csv file

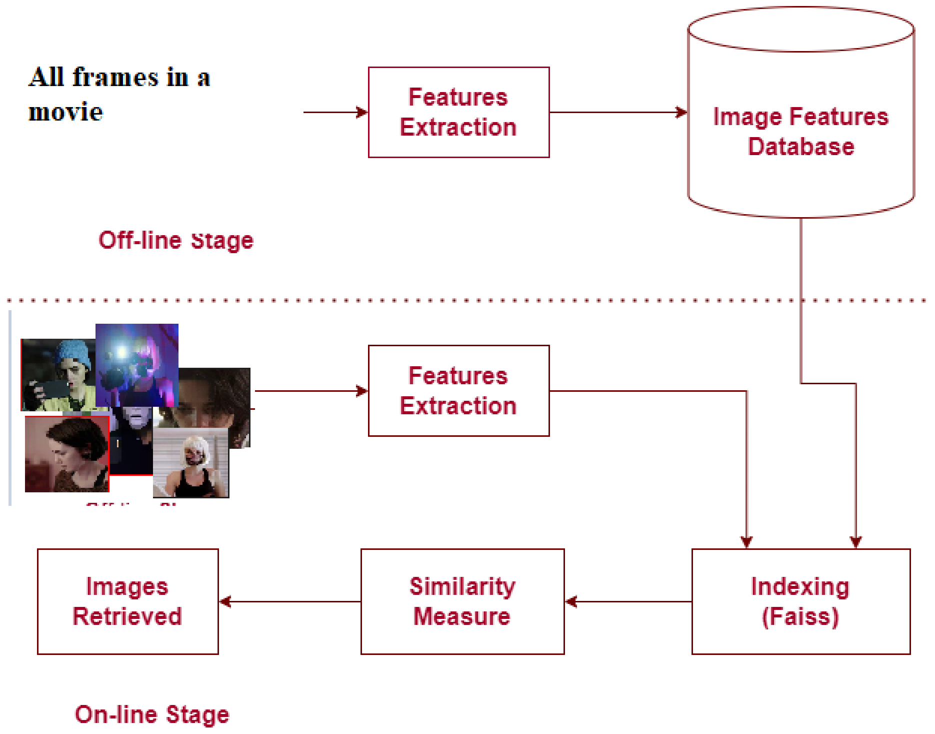


Person	Shot	Scene	Full
0	1	1	losing_ground-1-shot_1
1	1	1	losing_ground-1-shot_1
2	1	1	losing_ground-1-shot_1
3	1	1	losing_ground-1-shot_1
4	1	1	losing_ground-1-shot_1
5	1	1	losing_ground-1-shot_1
6	1	1	losing_ground-1-shot_1
7	1	1	losing_ground-1-shot_1
8	1	1	losing_ground-1-shot_1
9	1	1	losing_ground-1-shot_1
10	1	1	losing_ground-1-shot_1
11	1	1	losing_ground-1-shot_1
12	1	1	losing_ground-1-shot_1
13	1	1	losing_ground-1-shot_1
14	1	1	losing_ground-1-shot_1

Example of a .csv file to map persons to corresponding shots

2.1.2. Online stage

- Detect persons from query images
- Extract and normalize feature vectors of these characters
- Apply a searching algorithm to find matching vectors
- Find corresponding shots of these vectors according to the .csv file above



Pipeline of the retrieval system

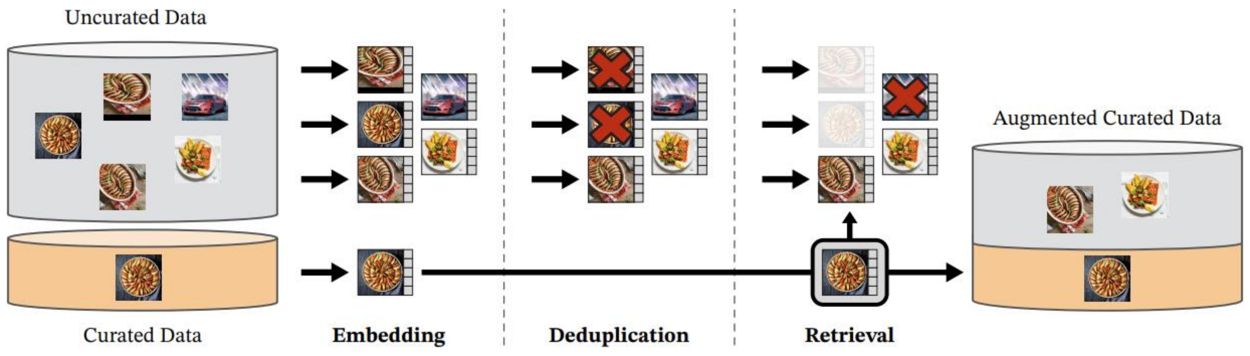
2.2. Methods used for these two stages

2.2.1. Person detection: YOLOv8-l

- YOLOv8 is the latest iteration in the YOLO series of real-time object detectors, offering cutting-edge performance in terms of accuracy and speed. Building upon the advancements of previous YOLO versions, YOLOv8 introduces new features and optimizations that make it an ideal choice for various object detection tasks in a wide range of applications.
- The YOLOv8 series offers a diverse range of models, each specialized for specific tasks in computer vision. These models are designed to cater to various requirements, from object detection to more complex tasks like instance segmentation, pose/keypoints detection, oriented object detection, and classification
- In this project, we utilize YOLOv8-l for person detection in frames and in query images. The number of parameters of this model is ~43.7 million.

2.2.2. Feature extraction: DINOv2

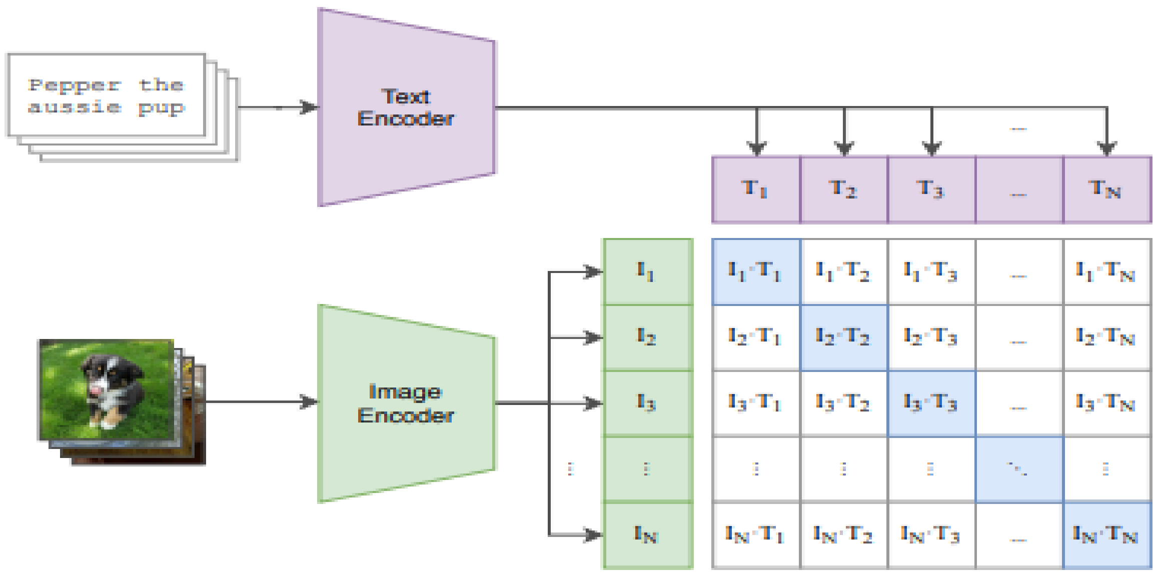
- DINOv2 is a pre-trained visual model with different Vision Transformers architectures. It is proposed by Maxim et al. , who are researchers at Meta AI, in 2023.
- It can learn visual features of images that can be used for various tasks: classification, semantic segmentation, instance retrieval, depth estimation, etc.
- DINOv2 is pretrained on curated and uncurated visual data. The curated data includes LVD-142M, comprised of ImageNet-22k, ImageNet-1k/train, Google Landmarks, Caltech-101/train, Food-101/train, etc. The uncurated data is crawled from a web repository that the team developing DINOv2 refuses to reveal.
- In the preprocessing phase, at first uncurated data is deduplicated. Then the curated data is augmented by being matched with uncurated data. The resulting combination augments the initial dataset through a self-supervised retrieval system. In order to do this, the authors first compute an image embedding using a self-supervised ViT-H/16 network pretrained on ImageNet-22k, and use cosine-similarity as a distance measure between images. Then, we perform k-means clustering of the uncurated data. Given a query dataset for retrieval, if it is large enough we retrieve N (typically 4) nearest neighbors for each query image. If it is small, we sample M images from the cluster corresponding to each query image. The authors adjust N and M by visual inspection of the retrieval result.
- The deduplication and retrieval stages of our pipeline rely on the FAISS library to efficiently index and compute batch searches of nearest embeddings.
- There are 3 versions of DINOv2: dinov2-large, dinov2-base and dinov2-small. With limited computing resources, in this project, dinov2-small is utilized. This version has ~22.1 million parameters and the embedding size of (257, 384). Note that the embedding is still large so each of its column is replaced with its mean value. Therefore, the new embedding would have a size of (1, 384).



DINOv2's preprocessing pipeline

2.2.3. Feature extraction: CLIP

- CLIP stands for Contrastive Language-Image Pre-Training, proposed by Alec et.al in 2021.
- It is trained on publicly available image-caption data sourced from a variety of websites and established image datasets like YFCC100M. A significant portion of the dataset was obtained through web crawling, resulting in a representation that leans towards demographics connected to the internet—primarily from more developed nations and with a bias towards younger, male users.
- It can use for various task like zero-shot, arbitrary image classification



Pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 … T_N
I_1 I_2 I_3 ⋮ I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ … $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ … $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ … $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ … $I_N \cdot T_N$

CLIP's constrative pre-learning phase

2.2.4. Constructing index for each movie: FAISS

- Facebook AI Similarity Search (FAISS) allows for efficient search of vectorized multimedia documents similar to a given set of vectorized query documents
- Vectors can be compared using L2 (Euclidean) distances or dot products
- Given query vectors and an index of vectorized documents, the document that is most similar to query vectors has lowest L2 or highest normalized dot product (cosine)
- In this project, FlatL2 metric is used. Its formula is as follows:

$$L2(u,v) = \sqrt{\sum_{i=1}^{m}(u_i - v_i)^2}$$

in which u and v are 2 vectors of the same length m

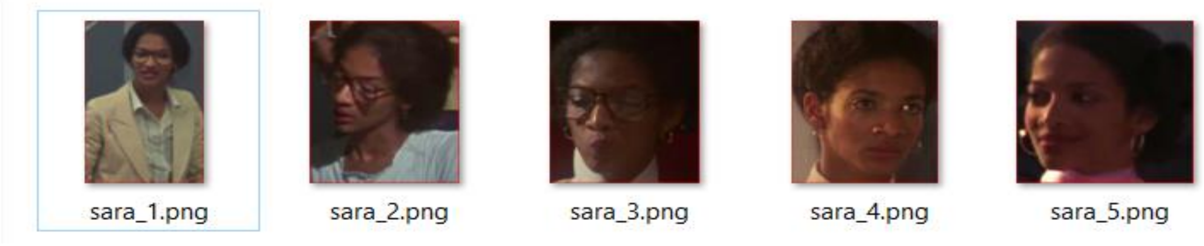
In FAISS, the corresponding index of FlatL2 is IndexFlatL2

III. EXPERIMENT

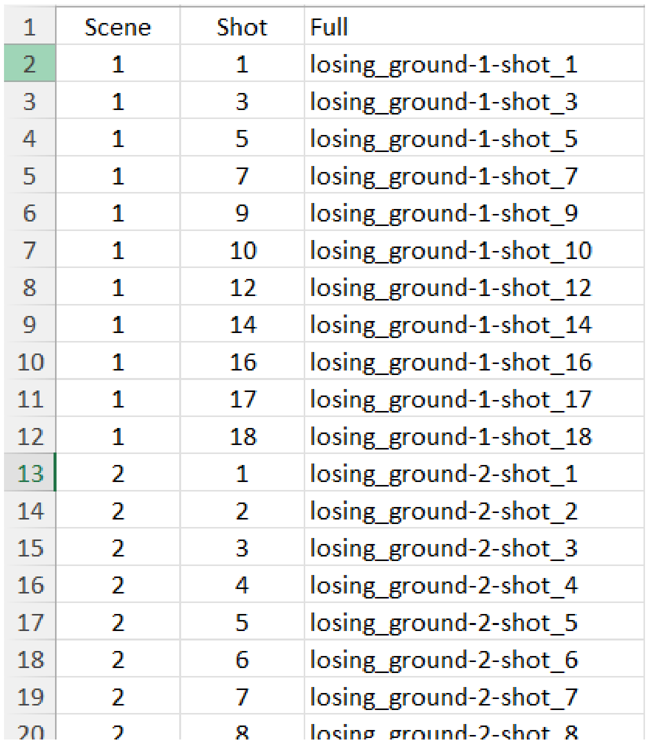
3.1. Dataset: TRECVID_MSUM_2022

Movie	No. scenes	No. shots	No. frames
Like Me	28	870	22440
Memphis	47	259	21650
Losing Ground	40	302	22600
Liberty Kid	56	1048	31803
Calloused Hands	58	965	26774

Each character in a movie also has a set of images used for querying. For example in Losing Ground, Sara has 5 query images as follows:



And here is the ground truth, that means the actual shots relevant to the query images:



1	Scene	Shot	Full
2	1	1	losing_ground-1-shot_1
3	1	3	losing_ground-1-shot_3
4	1	5	losing_ground-1-shot_5
5	1	7	losing_ground-1-shot_7
6	1	9	losing_ground-1-shot_9
7	1	10	losing_ground-1-shot_10
8	1	12	losing_ground-1-shot_12
9	1	14	losing_ground-1-shot_14
10	1	16	losing_ground-1-shot_16
11	1	17	losing_ground-1-shot_17
12	1	18	losing_ground-1-shot_18
13	2	1	losing_ground-2-shot_1
14	2	2	losing_ground-2-shot_2
15	2	3	losing_ground-2-shot_3
16	2	4	losing_ground-2-shot_4
17	2	5	losing_ground-2-shot_5
18	2	6	losing_ground-2-shot_6
19	2	7	losing_ground-2-shot_7
20	2	8	losing_ground-2-shot_8

3.2. Metrics

To evaluate the retrieval system, precision and recall are applied and calculated as follows:

$$Precision = \frac{number\ of\ true\ retrieved\ \boldsymbol{shots}}{number\ of\ \boldsymbol{retrieved\ shots}}$$

$$Recall = \frac{number\ of\ true\ retrieved\ \boldsymbol{shots}}{number\ of\ \boldsymbol{relevant\ shots}}$$

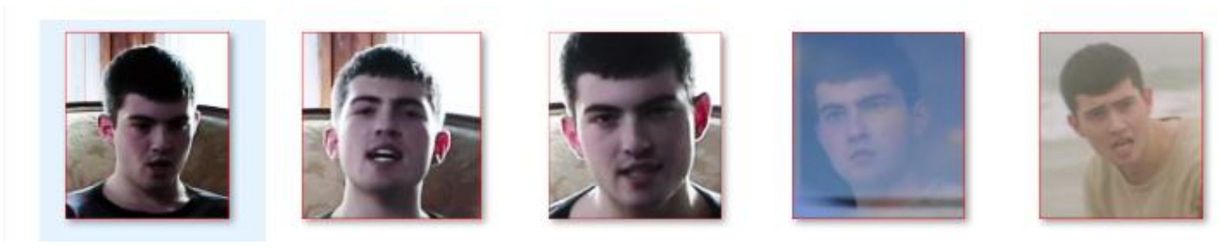
Recall that in this project, actually the "person" objects are retrieved. Only when they are matched with the corresponding do the shots be retrieved. Therefore, the retrieval system will be evaluated against $k = top\ number\ of\ most\ relevant\ persons\ in\ the\ movie$.

Recall that this project aims to compare the performance of DINOv2 and CLIP features in the retrieval system.

3.3. Evaluation

Firstly, experiment is conducted with two characters: Kiya and Burt from the Like Me (2017) film. Here are the query images of them:

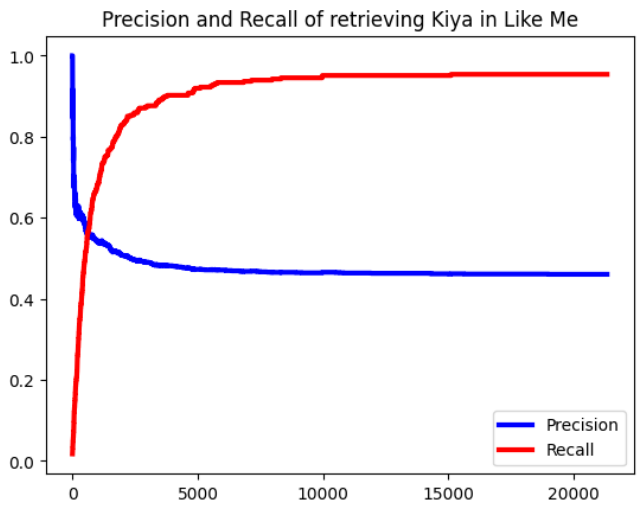
Kiya

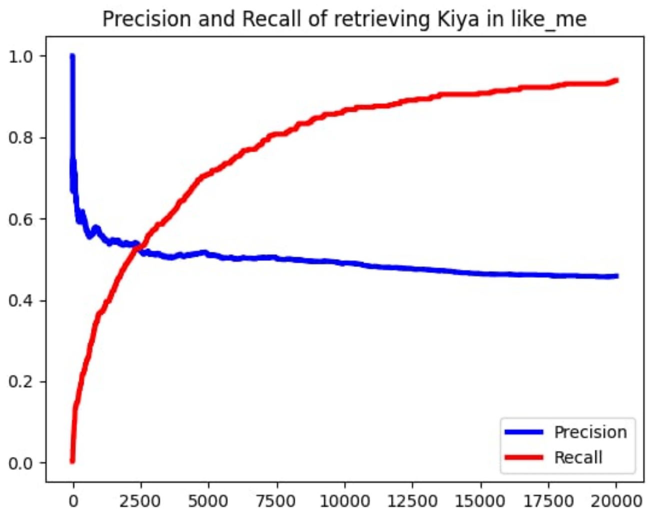


Burt

Here are two charts representing the performance of DINOv2 and CLIP when being used in retrieving Kiya in Like Me. The horizontal axis represents different values of k, top number of most relevant persons to Kiya. As can be seen below, precision and recall starts at ~1 and ~0, respectively. When k increases, they progress in contrastive trends. While there is an incredible increase in recall, the precision sees a notable decline. However, the changes are mainly taken place in a first few hundreds of k because after that both two values are almost unchanged.

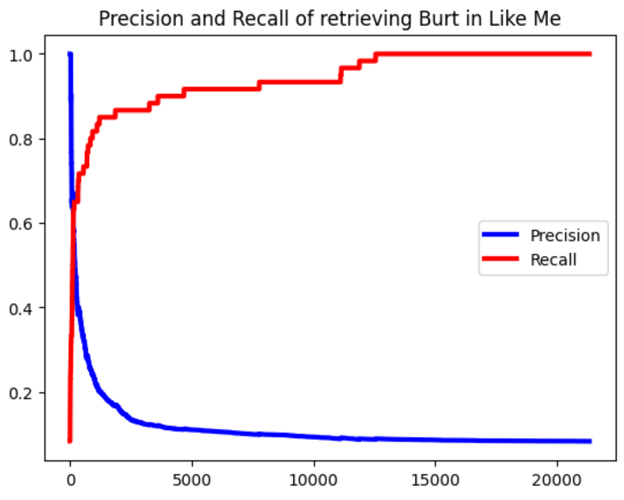
DINOv2 **CLIP**

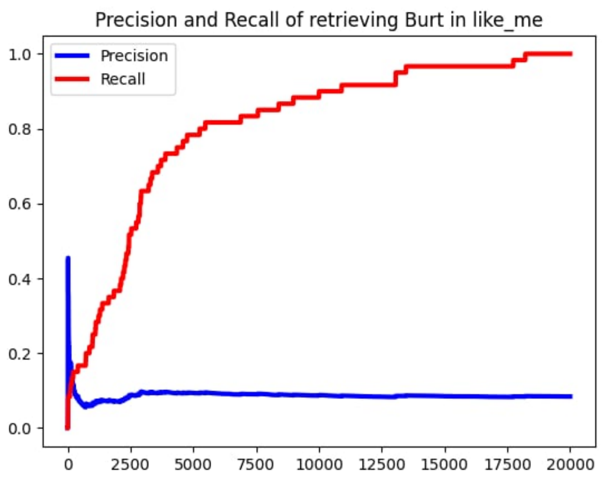




The charts for Burt character have precision and recall progressing in almost the same trends. However, looking at the DINOv2's chart, one can observe that the changes are very sharp at first few hundreds of k. In the CLIP's chart, there is a notable short sudden increase followed by a sharp decline in precision at first. In addition, the recall' line in both two charts is rough.

DINOv2 **CLIP**





The charts of other characters closely represents precision and recall as in the charts above. Please refer to pages 28 to 34 in the Powerpoint presentation.

IV. A DEMO APPLICATION

4.1. Streamlit

- Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps. It is a Python-based library specifically designed for machine learning engineers. Streamlit allows you to create a stunning-looking application with only a few lines of code.
- In this project, the retrieval system is embedded in a Streamlit app. The app allows users to select an available movie (the movie which has its index built) and choose available images for query. After running the retrieval system, the app will display retrieved shots and users can play them.

4.2. Components of the Streamlit app

- Film and Character Selection:
    - Users can choose a film from a predefined list and specify the character of interest.
    - Available films include "Like Me," "Calloused Hands," "Liberty Kid," "Losing Ground," and "Memphis."
- Top k Character Retrieval:
    - The application allows users to define the value of k, representing the number of most relevant characters to retrieve within the selected movie.
- Video Output Customization:
    - Users have the flexibility to choose the desired number of associated videos to be returned by the application.

V. CHALLENGES

- The project barely considers various image processing techniques. That is because plenty of frames in which the character does not have a clear appearance (too much illumination, blur, deformed face, masked face, etc.)
- The project barely considers situations in which the character’s info in the query set may be not enough (lack of images)
- Retrieving shots relevant to a specific query image

VI. TASK ASSIGNMENT

In this project, both Nguyen Nguyen Khoi and Do Minh Khoi are responsible for detecting persons from frames, calculating and normalizing feature vectors, constructing indexes for movies and carryiout evaluation on characters in the TRECVID_MSUM_2022. Regarding calculating feature vectors, Nguyen Nguyen Khoi utilizes DINOv2 while Do Minh Khoi sticks with CLIP. Streamlit app is programmed by Do Minh Khoi. Presentation slides are designed by Nguyen Nguyen Khoi.

VII. REFERENCES

[1] DINOv2: Learning Robust Visual Features without Supervision
[2] Hugging face: dinov2-small
[3] Learning Transferable Visual Models From Natural Language Supervision
[4] Hugging face : clip-vit-base-patch16
[5] Faiss: A library for efficient similarity search
[6] Image similarity with DINOv2 and FAISS
[7] Introduction to Facebook AI Similarity Search (Faiss)
[8] Python Tutorial: Streamlit
[9] Build Web App instantly for Machine Learning using Streamlit